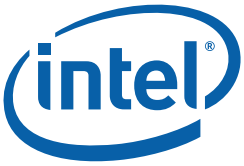

Intel® Pentium® Dual-Core Mobile Processor

Specification Update

February 2007

Document Number: 316515-001

Contents

Revision History

Revision	Description	Date
-001	Initial release	February 2007

§

Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents table. It is a compilation of device and document errata and specification clarifications and changes, and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

Related Documents

Document Title	Document Number/Location
Debug Port Design Guide for Crestline and Intel® 945PM/GM/GT and 940GML Express Chipset Systems	Note
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture	253665
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M	253666
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z	253667
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide	253668
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide	253669
Intel® 64 and IA-32 Architectures Optimization Reference Manual	248966
Intel® 64 and IA-32 Architectures Optimization Reference Manual Documentation Changes	252046

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics (e.g., core speed, L2 cache size, package type, etc.) as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number

Errata are design defects or errors. Errata may cause the Intel® Pentium® Dual-Core processor on 65-nm process behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

Note: Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

§

Summary Tables of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes, which apply to the listed Processor steppings. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or Specification Changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X: Erratum, Specification Change or Clarification that applies to this stepping.

(No mark) or (Blank Box): This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Status

Doc:	Document change or update that will be implemented.
Plan Fix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
No Fix:	There are no plans to fix this erratum.
Shaded:	This item is either new or modified from the previous version of the document.

Each Specification Update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Dual-Core Intel® Xeon® processor 7000 Δ sequence (Paxville MP)
B = Mobile Intel® Pentium® II processor
C = Intel® Celeron® processor
D = Dual-Core Intel® Xeon® processor 2.80 GHz
E = Intel® Pentium® III processor
F = Intel® Pentium® processor Extreme Edition and Intel® Pentium® D processor
G = Intel® Pentium® III Xeon® processor
H = Mobile Intel® Celeron® processor at 466 MHz, 433 MHz, 400 MHz, 366 MHz, 333 MHz, 300 MHz, and 266 MHz
I = Dual-Core Intel® Xeon® processor 5000 Δ series
J = 64-bit Intel® Xeon® processor MP with 1-MB L2 cache
K = Mobile Intel® Pentium® III processor – M
L = Intel® Celeron® D processor
M = Mobile Intel® Celeron® processor
N = Intel® Pentium® 4 processor
O = Intel® Xeon® processor MP
P = Intel® Xeon® processor
Q = Mobile Intel® Pentium® 4 processor supporting Hyper-Threading Technology on 90-nm process technology
R = Intel® Pentium® 4 processor on 90-nm process
S = 64-bit Intel® Xeon® processor with 800 MHz system bus
T = Mobile Intel® Pentium® 4 processor – M
U = Unannounced 64-bit Intel® Xeon® processor MP
V = Mobile Intel® Celeron® processor on 0.13 micron process in micro-FCPGA package
W = Intel® Celeron® M processor
X = Intel® Pentium® M processor on 90-nm process with 2-MB L2 cache
Y = Intel® Pentium® M processor
Z = Mobile Intel® Pentium® 4 processor with 533 MHz system bus
AA = Intel® Pentium® processor Extreme Edition and Intel® Pentium® D processor on 65-nm process
AB = Intel® Pentium® 4 processor on 65-nm process
AC = Intel® Celeron® Processor in 478 pin package
AD = Intel® Celeron® D processor on 65-nm process
AE = Intel® Core™ Duo processor and Intel® Core™ Solo processor on 65-nm process
AF = Dual-Core Intel® Xeon® processor LV
AG = Dual-Core Intel® Xeon® processor 5100 Δ series
AH = Intel® Core™2 Duo mobile processor
AI = Intel® Core™2 Extreme processor X6800 Δ and Intel® Core™2 Duo desktop processor E6000 Δ sequence
AJ = Quad-Core Intel® Xeon® processor 5300 Δ series

AK = Intel® Core™2 Extreme quad-core processor QX6700Δ, Intel® Core™2 quad processor Q6600 Δ and Quad-Core Intel® Xeon® processor 3200 Δ series
AL = Dual-Core Intel® Xeon® processor 7100 Δ series
AN = Intel® Pentium® Dual-Core mobile processor
AO = Quad-Core Intel® Xeon® processor 3200 Δ series
AP = Dual-Core Intel® Xeon® processor 3000 Δ series

Note: Δ Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

Number	D0	Plans	ERRATA
AN1	X	No Fix	FST Instruction with Numeric and Null Segment Exceptions May Take Numeric Exception with Incorrect FPU Operand Pointer
AN2	X	No Fix	Code Segment Limit Violation May Occur on 4-Gbyte Limit Check
AN3	X	No Fix	REP MOVS/STOS Executing with Fast Strings Enabled and Crossing Page Boundaries with Inconsistent Memory Types May Use an Incorrect Data Size or Lead to Memory-Ordering Violations
AN4	X	No Fix	Memory Aliasing with Inconsistent A and D Bits May Cause Processor Deadlock
AN5	X	No Fix	VM Bit Is Cleared on Second Fault Handled by Task Switch from Virtual-8086 (VM86)
AN6	X	No Fix	Page with PAT (Page Attribute Table) Set to USWC (Uncacheable Speculative Write Combine) While Associated MTRR (Memory Type Range Register) Is UC (Uncacheable) May Consolidate to UC
AN7	X	No Fix	FPU Operand Pointer May Not Be Cleared Following FINIT/FNINIT
AN8	X	No Fix	LTR Instruction May Result in Unexpected Behavior
AN9	X	No Fix	Invalid Entries in Page-Directory-Pointer-Table Register (PDPTR) May Cause General Protection (#GP) Exception If the Reserved Bits Are Set to One
AN10	X	No Fix	VMCALL When Executed during VMX Root Operation While CPL > 0 May Not Generate #GP Fault
AN11	X	No Fix	FP Inexact-Result Exception Flag May Not Be Set
AN12	X	No Fix	A Locked Data Access That Spans ccross Two Pages May Cause the System to Hang
AN13	X	No Fix	MOV to/from Debug Registers Causes Debug Exception
AN14	X	No Fix	INIT Does Not Clear Global Entries in the TLB
AN15	X	No Fix	Use of Memory Aliasing with Inconsistent Memory Type May Cause System Hang
AN16	X	No Fix	Machine Check Exception May Occur When Interleaving Code Between Different Memory Types
AN17	X	No Fix	Data Prefetch Performance Monitoring Events Can Only Be Enabled on a Single Core
AN18	X	No Fix	LOCK# Asserted during a Special Cycle Shutdown Transaction May Unexpectedly Deassert
AN19	X	No Fix	Disable Execution-Disable Bit (IA32_MISC_ENABLES [34]) Is Shared between Cores

Number	D0	Plans	ERRATA
AN20	X	No Fix	Last Branch Records (LBR) Updates May Be Incorrect after a Task Switch
AN21	X	No Fix	Address Reported by Machine-Check Architecture (MCA) on Single-bit L2 ECC Errors May Be Incorrect
AN22	X	No Fix	Disabling of Single-step On Branch Operation May Be Delayed following a POPFD Instruction
AN23	X	No Fix	Performance Monitoring Counters That Count External Bus Events May Report Incorrect Values after Processor Power State Transitions
AN24	X	No Fix	VERW/VERR/LSL/LAR Instructions May Unexpectedly Update the Last Exception Record (LER) MSR
AN25	X	No Fix	General Protection (#GP) Fault May Not Be Signaled on Data Segment Limit Violation above 4-G Limit
AN26	X	No Fix	Performance Monitoring Events for Retired Floating Point Operations (C1h) May Not Be Accurate
AN27	X	No Fix	DR3 Address Match on MOVD/MOVQ/MOVNTQ Memory Store Instruction May Incorrectly Increment Performance Monitoring Count for Saturating SIMD Instructions Retired (Event CFH)
AN28	X	No Fix	Global Pages in the Data Translation Look-Aside Buffer (DTLB) May Not Be Flushed by RSM Instruction before Restoring the Architectural State from SMRAM
AN29	X	No Fix	Data Breakpoint/Single Step on MOV SS/POP SS May Be Lost after Entry into SMM
AN30	X	No Fix	CS Limit Violation on RSM May Be Serviced before Higher Priority Interrupts/Exceptions
AN31	X	No Fix	Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced before Higher Priority Interrupts
AN32	X	No Fix	Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts
AN33	X	No Fix	The Processor May Report a #TS Instead of a #GP Fault
AN34	X	No Fix	BTS Message May Be Lost when the STPCLK# Signal Is Active
AN35	X	No Fix	Certain Performance Monitoring Counters Related to Bus, L2 Cache and Power Management Are Inaccurate
AN36	X	No Fix	A Write to an APIC Register Sometimes May Appear to Have Not Occurred
AN37	X	No Fix	IO_SMI Indication in SMRAM State Save Area May Be Set Incorrectly
AN38	X	No Fix	Simultaneous Access to the Same Page Table Entries by Both Cores May Lead to Unexpected Processor Behavior
AN39	X	No Fix	Logical Processors May Not Detect Write-Back (WB) Memory Writes
AN40	X	No Fix	Last Exception Record (LER) MSRs May be Incorrectly Updated
AN41	X	No Fix	SYSENTER/SYSEXIT Instructions Can Implicitly Load “Null Segment Selector” to SS and CS Registers
AN42	X	No Fix	Writing the Local Vector Table (LVT) when an Interrupt Is Pending May Cause an Unexpected Interrupt

Number	D0	Plans	ERRATA
AN43	X	No Fix	Using 2-M/4-M Pages When A20M# Is Asserted May Result in Incorrect Address Translations
AN44	X	No Fix	Counter Enable Bit [22] of IA32_CR_PerfEvtSel0 and IA32_CR_PerfEvtSel1 Do Not Comply with PerfMon (Architectural Performance Monitoring) Specification
AN45	X	No Fix	Premature Execution of a Load Operation Prior to Exception Handler Invocation
AN46	X	No Fix	Performance Monitoring Events for Retired Instructions (C0H) May Not Be Accurate
AN47	X	No Fix	#GP Fault is Not Generated on Writing IA32_MISC_ENABLE [34] When Execute Disable Bit Is Not Supported
AN48	X	No Fix	Update of Read/Write (R/W) or User/Supervisor (U/S) or Present (P) Bits without TLB Shootdown May Cause Unexpected Processor Behavior
AN49	X	No Fix	SSE/SSE2 Streaming Store Resulting in a Self-Modifying Code (SMC) Event May Cause Unexpected Behavior
AN50	X	No Fix	Shutdown Condition May Disable Non-Bootstrap Processors
AN51	X	No Fix	Split Locked Stores May Not Trigger the Monitoring Hardware
AN52	X	No Fix	Writing Shared Unaligned Data that Crosses a Cache Line without Proper Semaphores or Barriers May Expose a Memory Ordering Issue
AN53	X	No Fix	MSRs Actual Frequency Clock Count (IA32_APERF) or Maximum Frequency Clock Count (IA32_MPERF) May Contain Incorrect Data after a Machine Check Exception (MCE)
AN54	X	No Fix	Using Memory Type Aliasing with Memory Types WB/WT May Lead to Unpredictable Behavior
AN55	X	No Fix	Code Breakpoint May Be Taken after POP SS Instruction if It Is Followed by an Instruction That Faults
AN56	X	No Fix	Incorrect Address Computed for Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update
AN57	X	No Fix	Values for LBR/BTS/BTM Will Be Incorrect after an Exit from SMM
AN58	X	No Fix	EFLAGS Discrepancy on a Page Fault after a Multiprocessor TLB Shootdown
AN59	X	No Fix	Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior
AN60	X	No Fix	A Thermal Interrupt Is Not Generated when the Current Temperature Is Invalid
AN61	X	No Fix	Performance Monitoring Event FP_ASSIST May Not Be Accurate
AN62	X	No Fix	The BS Flag in DR6 May Be Set for Non-Single-Step #DB Exception
AN63	X	No Fix	BTM/BTS Branch-From Instruction Address May Be Incorrect for Software Interrupts
AN64	X	No Fix	Store to WT Memory Data May Be Seen in Wrong Order by Two Subsequent Loads
AN65	X	No Fix	Single Step Interrupts with Floating Point Exception Pending May Be Mishandled
AN66	X	No Fix	Fault on ENTER Instruction May Result in Unexpected Values on Stack Frame
AN67	X	No Fix	Non-Temporal Data Store May Be Observed in Wrong Program Order
AN68	X	No Fix	Unaligned Accesses to Paging Structures May Cause the Processor to Hang

Number	D0	Plans	ERRATA
AN69	X	No Fix	Microcode Updates Performed during VMX Non-root Operation Could Result in Unexpected Behavior
AN70	X	No Fix	INVLPG Operation for Large (2-M/4-M) Pages May Be Incomplete Under Certain Conditions
AN71	X	No Fix	Page Access Bit May Be Set Prior to Signaling a Code Segment Limit Fault
AN72	X	No Fix	Performance Monitoring Events for Hardware Prefetch Requests (4EH) and Hardware Prefetch Request Cache Misses (4FH) May Not be Accurate
AN73	X	No Fix	EFLAGS, CR0, CR4 and the EXF4 Signal May Be Incorrect after Shutdown
AN74	X	No Fix	An Asynchronous MCE during a Far Transfer May Corrupt ESP
AN75	X	No Fix	Store Ordering May Be Incorrect between WC and WP Memory Types

Number	SPECIFICATION CHANGES
	There are no Specification Changes in this Specification Update revision.

Number	SPECIFICATION CLARIFICATIONS
	There are no Specification Clarifications in this Specification Update revision.

Number	DOCUMENTATION CHANGES
	There are no Documentation Changes in this Specification Update revision.

§

Identification Information

Component Marking Information

Figure 1. Intel® Pentium® Dual-Core Mobile Processor on 65-nm Process (Micro-FCPGA/FCBGA) S-Spec Markings

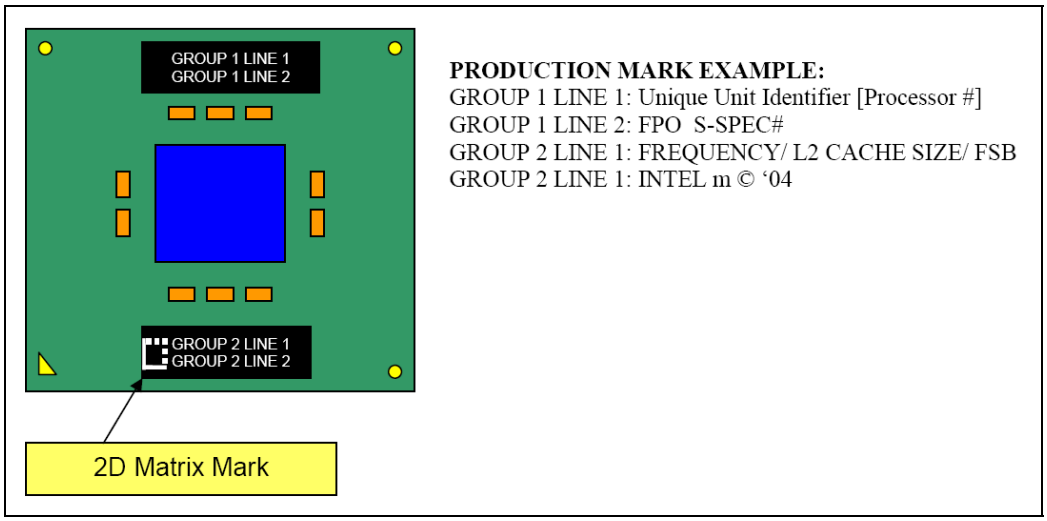

Table 1. Pentium Dual-Core Mobile Processor on 65-nm Process Identification Information

QDF/S-SPEC#	Processor #	Package	Stepping	CPUID	FSB(MHz)	Speed HFM/LFM (GHz)	Notes
SL9VX	T2060	Micro-FCPGA	D-0	06ECh	533	1.6/800	1

NOTES:
1. VCC_CORE=1.2125 V-1.025 V for HFM Range/LFM.

§

Errata

AN1. FST Instruction with Numeric and Null Segment Exceptions May Take Numeric Exception with Incorrect FPU Operand Pointer

Problem: If execution of an FST (Store Floating Point Value) instruction would generate both numeric and Null segment exceptions, the numeric exceptions may be taken first and with the Null x87 FPU Instruction Operand (Data) Pointer.

Implication: Due to this erratum, on an FST instruction the processor reports a numeric exception instead of reporting an exception because of a Null segment. If the numeric exception handler tries to access the FST data it will get a #GP fault. Intel has not observed this erratum with any commercially available software, or system.

Workaround: The numeric exception handler should check the segment and if it is Null avoid further access to the data that caused the fault.

Status: For the steppings affected, see the Summary Tables of Changes.

AN2. Code Segment Limit Violation May Occur on 4-Gbyte Limit Check

Problem: Code Segment limit violation may occur on 4-Gbyte limit check when the code stream wraps around in a way that one instruction ends at the last byte of the segment and the next instruction begins at 0x0.

Implication: This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

Workaround: Avoid code that wraps around segment limit.

Status: For the steppings affected, see the Summary Tables of Changes.

AN3. REP MOVS/STOS Executing with Fast Strings Enabled and Crossing Page Boundaries with Inconsistent Memory Types May Use an Incorrect Data Size or Lead to Memory-Ordering Violations

Problem: Under certain conditions as described in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A,* Section 7.2.3 *Out-of-Order Stores For String Operations in Pentium 4, Intel Xeon, and P6 Family Processors*, the processor performs REP MOVS or REP STOS as fast strings. Due to this erratum fast string REP MOVS/REP STOS instructions that cross page boundaries from WB/WC memory types to UC/WP/WT memory types, may start using an incorrect data size or may observe memory ordering violations.

Implication: Upon crossing the page boundary the following may occur, dependent on the new page memory type:

- UC the data size of each write will now always be 8 bytes, as opposed to the original data size.
- WP the data size of each write will now always be 8 bytes, as opposed to the original data size and there may be a memory ordering violation.
- WT there may be a memory ordering violation.

Workaround: Software should avoid crossing page boundaries from WB or WC memory type to UC, WP or WT memory type within a single REP MOVS or REP STOS instruction that will execute with fast strings enabled.

Status: For the steppings affected, see the Summary Tables of Changes.

AN4. Memory Aliasing with Inconsistent A and D Bits May Cause Processor Deadlock

Problem: In the event that software implements memory aliasing by having two Page Directory Entries (PDEs) point to a common Page Table Entry (PTE) and the Accessed and Dirty bits for the two PDEs are allowed to become inconsistent the processor may become deadlocked.

Implication: This erratum has not been observed with commercially available software.

Workaround: Software that needs to implement memory aliasing in this way should manage the consistency of the Accessed and Dirty bits.

Status: For the steppings affected, see the Summary Tables of Changes.

AN5. VM Bit is Cleared on Second Fault Handled by Task Switch from Virtual-8086 (VM86)

Problem: Following a task switch to any fault handler that was initiated while the processor was in VM86 mode, if there is an additional fault while servicing the original task switch then the VM bit will be incorrectly cleared in EFLAGS, data segments will not be pushed and the processor will not return to the correct mode upon completion of the second fault handler via IRET.

Implication: When the OS recovers from the second fault handler, the processor will no longer be in VM86 mode. Normally, operating systems should prevent interrupt task switches from faulting, thus the scenario should not occur under normal circumstances.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN6. Page with PAT (Page Attribute Table) Set to USWC (Uncacheable Speculative Write Combine) While Associated MTRR (Memory Type Range Register) Is UC (Uncacheable) May Consolidate to UC

Problem: A page whose PAT memory type is USWC while the relevant MTRR memory type is UC, the consolidated memory type may be treated as UC (rather than WC, as specified in the *Intel® 64 and IA-32 Architectures Software Developer's Manuals*).

Implication: When this erratum occurs, the memory page may be as UC (rather than WC). This may have a negative performance impact.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN7. FPU Operand Pointer May Not Be Cleared Following FINIT/FNINIT

Problem: Initializing the floating point state with either FINIT or FNINT, may not clear the x87 FPU Operand (Data) Pointer Offset and the x87 FPU Operand (Data) Pointer Selector (both fields form the FPUDataPointer). Saving the floating point environment with FSTENV, FNSTENV, or floating point state with FSAVE, FNSAVE or FXSAVE before an intervening FP instruction may save un-initialized values for the FPUDataPointer.

Implication: When this erratum occurs, the values for FPUDataPointer in the saved floating point image structure may appear to be random values. Executing any non-control FP instruction with memory operand will initialize the FPUDataPointer. Intel has not observed this erratum with any commercially available software.

Workaround: After initialization, do not expect a floating point state saved memory image to be correct, until at least one non-control FP instruction with a memory operand has been executed.

Status: For the steppings affected, see the Summary Tables of Changes.

AN8. LTR Instruction May Result in Unexpected Behavior

Problem: Under certain circumstances an LTR (Load Task Register) instruction may result in an unexpected behavior if all the following conditions are met:

1. Invalid data selector of the TR (Task Register) resulting with either #GP (General Protection Fault) or #NP (Segment Not Present Fault).
2. GDT (Global Descriptor Table) is not 8-bytes aligned.

Implication: If all conditions have been met then under certain circumstances LTR instruction may result in system hang, memory corruption or other unexpected behavior. This erratum has not been observed in commercial operating systems or software.

Workaround: Operating system software should align GDT to 8-bytes, as recommended in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A* section 3.5.1 "Segment Descriptor Tables". For performance reasons, GDT is typically aligned to 8-bytes.

Status: For the steppings affected, see the Summary Tables of Changes.

AN9. Invalid Entries in Page-Directory-Pointer-Table Register (PDPTR) May Cause General Protection (#GP) Exception If the Reserved Bits Are Set to One

Problem: Invalid entries in the Page-Directory-Pointer-Table Register (PDPTR) that have the reserved bits set to one may cause a General Protection (#GP) exception.

Implication: Intel has not observed this erratum with any commercially available software.

Workaround: Do not set the reserved bits to one when PDPTR entries are invalid.

Status: For the steppings affected, see the Summary Tables of Changes.

AN10. VMCALL When Executed during VMX Root Operation While CPL > 0 May Not Generate #GP Fault

Problem: If VMCALL is executed during VMX root operation with CPL > 0, the expected behavior is for the processor to generate a General Protection Fault (#GP). Due to this erratum, the #GP fault may not be generated.

Implication: VM Monitor code running with CPL > 0 may not generate #GP fault on VMCALL, but still will behave as if VM Exit had occurred.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN11. FP Inexact-Result Exception Flag May Not Be Set

Problem: When the result of a floating-point operation is not exactly represented in the destination format (1/3 in binary form, for example), an inexact-result (precision) exception occurs. When this occurs, the PE bit (bit 5 of the FPU status word) is normally set by the processor. Under certain rare conditions, this bit may not be set when this rounding occurs. However, other actions taken by the processor (invoking the software exception handler if the exception is unmasked) are not affected. This erratum can only occur if the floating-point operation which causes the precision exception is immediately followed by one of the following instructions:

- FST m32real
- FST m64real
- FSTP m32real
- FSTP m64real
- FSTP m80real
- FIST m16int
- FIST m32int
- FISTP m16int
- FISTP m32int
- FISTP m64int

Note that even if this combination of instructions is encountered, there is also a dependency on the internal pipelining and execution state of both instructions in the processor.

Implication: Inexact-result exceptions are commonly masked or ignored by applications, as it happens frequently, and produces a rounded result acceptable to most applications. The PE bit of the FPU status word may not always be set upon receiving an inexact-result exception. Thus, if these exceptions are unmasked, a floating-point error exception handler may not recognize that a precision exception occurred. Note that this is a "sticky" bit, i.e., once set by an inexact-result condition, it remains set until cleared by software.

Workaround: This condition can be avoided by inserting either three NOPs or three non-floating-point non-Jcc instructions between the two floating-point instructions.

Status: For the steppings affected, see the Summary Tables of Changes.

AN12. A Locked Data Access that Spans Across Two Pages May Cause the System to Hang

Problem: An instruction with lock data access that spans across two pages may, given some rare internal conditions, hang the system.

Implication: When this erratum occurs, the system may hang. Intel has not observed this erratum with any commercially available software or system.

Workaround: A locked data access should always be aligned.

Status: For the steppings affected, see the Summary Tables of Changes.

AN13. MOV To/From Debug Registers Causes Debug Exception

Problem: When in V86 mode, if a MOV instruction is executed to/from a debug register, a general-protection exception (#GP) should be generated. However, in the case when the general detect enable flag (GD) bit is set, the observed behavior is that a debug exception (#DB) is generated instead.

Implication: With debug-register protection enabled (i.e., the GD bit set), when attempting to execute a MOV on debug registers in V86 mode, a debug exception will be generated instead of the expected general-protection fault.

Workaround: In general, operating systems do not set the GD bit when they are in V86 mode. The GD bit is generally set and used by debuggers. The debug exception handler should check that the exception did not occur in V86 mode before continuing. If the exception did occur in V86 mode, the exception may be directed to the general-protection exception handler.

Status: For the steppings affected, see the Summary Tables of Changes.

AN14. INIT Does Not Clear Global Entries in the TLB

Problem: INIT may not flush a TLB entry when:

1. The processor is in protected mode with paging enabled and the page global enable flag is set (PGE bit of CR4 register)
2. G bit for the page table entry is set
3. TLB entry is present in TLB when INIT occurs

Implication: Software may encounter unexpected page fault or incorrect address translation due to a TLB entry erroneously left in TLB after INIT.

Workaround: Write to CR3, CR4 (setting bits PSE, PGE or PAE) or CR0 (setting bits PG or PE) registers before writing to memory early in BIOS code to clear all the global entries from TLB.

Status: For the steppings affected, see the Summary Tables of Changes.

AN15. Use of Memory Aliasing with Inconsistent Memory Type May Cause System Hang

Problem: Software that implements memory aliasing by having more than one linear addresses mapped to the same physical page with different cache types may cause the system to hang. This would occur if one of the addresses is non-cacheable used in code segment and the other a cacheable address. If the cacheable address finds its way in instruction cache, and non-cacheable address is fetched in IFU, the processor may invalidate the non-cacheable address from the fetch unit. Any micro-architectural event that causes instruction restart will expect this instruction to still be in fetch unit and lack of it will cause system hang.

Implication: This erratum has not been observed with commercially available software.

Workaround: Although it is possible to have a single physical page mapped by two different linear addresses with different memory types, Intel has strongly discouraged this practice as it may lead to undefined results. Software that needs to implement memory aliasing should manage the memory type consistency.

Status: For the steppings affected, see the Summary Tables of Changes.

AN16. Machine Check Exception May Occur When Interleaving Code between Different Memory Types

Problem: A small window of opportunity exists where code fetches interleaved between different memory types may cause a machine check exception. A complex set of micro-architectural boundary conditions is required to expose this window.

Implication: Interleaved instruction fetches between different memory types may result in a machine check exception. The system may hang if machine check exceptions are disabled. Intel has not observed the occurrence of this erratum while running commercially available applications or operating systems.

Workaround: Software can avoid this erratum by placing a serializing instruction between code fetches between different memory types.

Status: For the steppings affected, see the Summary Tables of Changes.

AN17. Data Prefetch Performance Monitoring Events Can Only Be Enabled on a Single Core

Problem: Current implementation of Data Prefetch performance monitoring events allows counting only for a single core at a time.

Implication: Dual core support for counting Data Prefetch performance monitoring events is not currently available.

Workaround: Software should enable Data Prefetch performance monitoring events on one core at a time.

Status: For the steppings affected, see the Summary Tables of Changes.

AN18. LOCK# Asserted during a Special Cycle Shutdown Transaction May Unexpectedly Deassert

Problem: During a processor shutdown transaction, when LOCK# is asserted and if a DEFER# is received during a snoop phase and the Locked transaction is pipelined on the front side bus (FSB), LOCK# may unexpectedly deassert.

Implication: When this erratum occurs, the system may hang during shutdown. Intel has not observed this erratum with any commercially available systems or software.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN19. Disable Execution-Disable Bit (IA32_MISC_ENABLES [34]) Is Shared between Cores

Problem: The bit 34 of the IA32_MISC_ENABLES Model Specific Register (MSR) is shared between the execution cores.

Implication: Both cores will operate according to the shared value of bit IA32_MISC_ENABLES [34].

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN20. Last Branch Records (LBR) Updates May Be Incorrect after a Task Switch

Problem: A Task-State Segment (TSS) task switch may incorrectly set the LBR_FROM value to the LBR_TO value.

Implication: The LBR_FROM will have the incorrect address of the Branch Instruction.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN21. Address Reported By Machine-Check Architecture (MCA) on Single-bit L2 ECC Errors May Be Incorrect

Problem: When correctable single-bit ECC errors occur in the L2 cache the address is logged in the MCA address register (MCi_ADDR). Under some scenarios, the address reported may be incorrect.

Implication: Software should not rely on the value reported in MCi_ADDR, for Single-bit L2 ECC errors

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN22. Disabling of Single-step on Branch Operation May Be Delayed Following a POPFD Instruction

Problem: Disabling of Single-step On Branch Operation may be delayed, if the following conditions are met:

"Single Step On Branch Mode" is enabled (DebugCtlMSR.BTF and EFLAGS.TF are set)

POPFD used to clear EFLAGS.TF

A jump instruction (JMP, Jcc, etc.) is executed immediately after POPFD

Implication: Single-step On Branch mode may remain in effect for one instruction after the POPFD instruction disables it by clearing the EFLAGS.TF bit.

Workaround: There is no workaround for Single-Step operation in commercially available software. The workaround for custom software is to execute at least one instruction following POPFD before issuing a JMP instruction.

Status: For the steppings affected, see the Summary Tables of Changes.

AN23. Performance Monitoring Counters that Count External Bus Events May Report Incorrect Values after Processor Power State Transitions

Problem: Performance monitoring counters that count external bus events operate when the processor is in the Active state (C0). If a processor transitions to a new power state, these Performance monitoring counters will stop counting, even if the event being counted remains active.

Implication: After transitioning between processor power states, software may observe incorrect counts in Performance monitoring counters that count external bus events.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN24. VERW/VERR/LSL/LAR Instructions May Unexpectedly Update the Last Exception Record (LER) MSR

Problem: The LER MSR may be unexpectedly updated, if the resultant value of the Zero Flag (ZF) is zero after executing the following instructions:

VERR (ZF=0 indicates unsuccessful segment read verification)

VERW (ZF=0 indicates unsuccessful segment write verification)

LAR (ZF=0 indicates unsuccessful access rights load)

LSL (ZF=0 indicates unsuccessful segment limit load)

Implication: The value of the LER MSR may be inaccurate if VERW/VERR/LSL/LAR instructions are executed after the occurrence of an exception.

Workaround: Software exception handlers that rely on the LER MSR value should read the LER MSR before executing VERW/VERR/LSL/LAR instructions.

Status: For the steppings affected, see the Summary Tables of Changes.

AN25. General Protection (#GP) Fault May Not Be Signaled on Data Segment Limit Violation above 4-G Limit

Problem: Memory accesses to flat data segments (base = 00000000h) that occur above the 4G limit (0ffffffffh) may not signal a #GP fault.

Implication: When such memory accesses occur, the system may not issue a #GP fault.

Workaround: Software should ensure that memory accesses do not occur above the 4G limit (0ffffffffh).

Status: For the steppings affected, see the Summary Tables of Changes.

AN26. Performance Monitoring Events for Retired Floating Point Operations (C1h) May Not Be Accurate

Problem: Performance monitoring events that count retired floating point operations may be too high.

Implication: The Performance Monitoring Event may have an inaccurate count.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN27. DR3 Address Match on MOVD/MOVQ/MOVNTQ Memory Store Instruction May Incorrectly Increment Performance Monitoring Count for Saturating SIMD Instructions Retired (Event CFh)

Problem: Performance monitoring for Event CFH normally increments on saturating SIMD instruction retired. Regardless of DR7 programming, if the linear address of a retiring memory store MOVD/MOVQ/MOVNTQ instruction executed matches the address in DR3, the CFH counter may be incorrectly incremented.

Implication: The value observed for performance monitoring count for saturating SIMD instructions retired may be too high. The size of error is dependent on the number of occurrences of the conditions described above, while the counter is active.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN28. Global Pages in the Data Translation Look-Aside Buffer (DTLB) May Not Be Flushed by RSM Instruction before Restoring the Architectural State from SMRAM

Problem: The Resume from System Management Mode (RSM) instruction does not flush global pages from the Data Translation Look-Aside Buffer (DTLB) prior to reloading the saved architectural state.

Implication: If SMM turns on paging with global paging enabled and then maps any of linear addresses of SMRAM using global pages, RSM load may load data from the wrong location.

Workaround: Do not use global pages in system management mode.

Status: For the steppings affected, see the Summary Tables of Changes.

AN29. Data Breakpoint/Single Step on MOV SS/POP SS May Be Lost after Entry into SMM

Problem: Data Breakpoint/Single Step exceptions are normally blocked for one instruction following MOV SS/POP SS instructions. Immediately after executing these instructions, if the processor enters SMM (System Management Mode), upon RSM (resume from SMM) operation, normal processing of Data Breakpoint/Single Step exceptions is restored.

Because of this erratum, Data Breakpoints/Single step exceptions on MOVSS/POPSS instructions may be lost under one of the following conditions.

- Following SMM entry and after RSM, the next instruction to be executed is HLT or MWAIT
- SMM entry after executing MOV SS/POP SS is the result of executing an I/O instruction that triggers a synchronous SMI (System Management Interrupt).

Implication: Data Breakpoints/Single step operation on MOV SS/POP SS instructions may be unreliable in the presence of SMI.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN30. CS Limit Violation on RSM May Be Serviced before Higher Priority Interrupts/Exceptions

Problem: When the processor encounters a CS (Code Segment) limit violation, a #GP (General Protection Exception) fault is generated after all higher priority Interrupts and exceptions are serviced. Because of this erratum, if RSM (Resume from System Management Mode) returns to execution flow where a CS limit violation occurs, the #GP fault may be serviced before a higher priority Interrupt or Exception (e.g., NMI (Non-Maskable Interrupt), Debug break(#DB), Machine Check (#MC), etc).

Implication: Operating systems may observe a #GP fault being serviced before higher priority interrupts and Exceptions.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN31. Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced before Higher Priority Interrupts

Problem: Interrupts that are pending prior to the execution of the STI (Set Interrupt Flag) instruction are serviced immediately after the STI instruction is executed. Because of this erratum, if following STI, an instruction that triggers a #MF is executed while STPCLK#, Enhanced Intel SpeedStep® Technology transitions or Thermal Monitor 1 events occur, the pending #MF may be serviced before higher priority interrupts.

Implication: Software may observe #MF being serviced before higher priority interrupts.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN32. Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts

Problem: Software can enable DTS thermal interrupts by programming the thermal threshold and setting the respective thermal interrupt enable bit. When programming DTS value, the previous DTS threshold may be crossed. This will generate an unexpected thermal interrupt.

Implication: Software may observe an unexpected thermal interrupt occur after reprogramming the thermal threshold.

Workaround: In the ACPI/OS implement a workaround by temporarily disabling the DTS threshold interrupt before updating the DTS threshold value.

Status: For the steppings affected, see the Summary Tables of Changes.

AN33. The Processor May Report a #TS Instead of a #GP Fault

Problem: A jump to a busy TSS (Task-State Segment) may cause a #TS (invalid TSS exception) instead of a #GP fault (general protection exception).

Implication: Operation systems that access a busy TSS may get invalid TSS fault instead of a #GP fault. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN34. BTS Message May Be Lost When the STPCLK# Signal Is Active

Problem: STPCLK# is asserted to enable the processor to enter a low-power state. Under some circumstances, when STPCLK# becomes active, a pending BTS (Branch Trace Store) message may be either lost and not written or written with corrupted branch address to the Debug Store area.

Implication: BTS messages may be lost in the presence of STPCLK# assertions.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN35. Certain Performance Monitoring Counters Related to Bus, L2 Cache and Power Management Are Inaccurate

Problem: All Performance Monitoring Counters in the ranges 21H-3DH and 60H-7FH may have inaccurate results up to ±7.

Implication: There may be a small error in the affected counts.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN36. A Write to an APIC Register Sometimes May Appear to Have Not Occurred

Problem: With respect to the retirement of instructions, stores to the uncacheable memory-based APIC register space are handled in a non-synchronized way. For example if an instruction that masks the interrupt flag, e.g., CLI, is executed soon after an uncacheable write to the Task Priority Register (TPR) that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR, but higher than the final TPR, to not be serviced until the interrupt enabled flag is finally set, i.e., by STI instruction. Interrupts will remain pending and are not lost.

Implication: In this example the processor may allow interrupts to be accepted but may delay their service.

Workaround: This non-synchronization can be avoided by issuing an APIC register read after the APIC register write. This will force the store to the APIC register before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AN37. IO_SMI Indication in SMRAM State Save Area May Be Set Incorrectly

Problem: The IO_SMI bit in SMRAM's location 7FA4H is set to "1" by the CPU to indicate a System Management Interrupt (SMI) occurred as the result of executing an instruction that reads from an I/O port. Due to this erratum, the IO_SMI bit may be incorrectly set by:

- A non-I/O instruction.
- SMI is pending while a lower priority event interrupts
- A REP I/O read
- An I/O read that redirects to MWAIT.
- In systems supporting Intel® Virtualization Technology a fault in the middle of an IO operation that causes a VM Exit

Implication: SMM handlers may get false IO_SMI indication.

Workaround: The SMM handler has to evaluate the saved context to determine if the SMI was triggered by an instruction that read from an I/O port. The SMM handler must not restart an I/O instruction if the platform has not been configured to generate a synchronous SMI for the recorded I/O port address.

Status: For the steppings affected, see the Summary Tables of Changes.

AN38. Simultaneous Access to the Same Page Translation Entries by Both Cores May Lead to Unexpected Processor Behavior

Problem: When the following conditions occur simultaneously, this may create a rare internal condition which may lead to unexpected processor behavior.

- One core is updating a page table entry, including the processor setting the Accessed and/or Dirty bits in the PTE as the result of an access
- The other core is using the same translation entry.

Implication: Unpredictable behavior in the processor may lead to livelock and shutdown. Intel has not observed this erratum with any commercially available software.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN39. Logical Processors May Not Detect Write-Back (WB) Memory Writes

Problem: Multiprocessor systems may use polling of memory semaphores to synchronize software activity. Because of this erratum, if a logical processor is polling a WB memory location while it is being updated by another logical processor, the update may not be detected.

Implication: System may livelock due to polling loop and undetected semaphore change. Intel has not observed this erratum on commercially available systems.

Workaround: It is possible for BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AN40. Last Exception Record (LER) MSRs May Be Incorrectly Updated

Problem: The LASTINTTOIP and LASTINTFROMIP MSRs (1DDH-1DEH) may contain incorrect values after the following events: masked SSE2 floating-point exception, StopClk, NMI and INT.

Implication: The value of the LER MSR may be incorrectly updated to point to a SIMD Floating-Point instruction even though no exception occurred on that instruction or to point to an instruction that was preceded by a StopClk interrupt or rarely not to be updated on Interrupts (NMI and INT).

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN41. SYSENTER/SYSEXIT Instructions Can Implicitly Load "Null Segment Selector" to SS and CS Registers

Problem: According to the processor specification, attempting to load a Null segment selector into the CS and SS segment registers should generate a General Protection Fault (#GP). Although loading a Null segment selector to the other segment registers is allowed, the processor will generate an exception when the segment register holding a Null selector is used to access memory. However, the SYSENTER instruction can implicitly load a Null value to the SS segment selector. This can occur if the value in SYSENTER_CS_MSR is between FFF8h and FFFBh when the SYSENTER instruction is executed. This behavior is part of the SYSENTER/SYSEXIT instruction definition; the content of the SYSTEM_CS_MSR is always incremented by 8 before it is loaded into the SS. This operation will set the Null bit in the segment selector if a Null result is generated, but it does not generate a #GP on the SYSENTER instruction itself. An exception will be generated as expected when the SS register is used to access memory, however. The SYSEXIT instruction will also exhibit this behavior for both CS and SS when executed with the value in SYSENTER_CS_MSR between FFF0h and FFF3h, or between FFE8h and FFEBh, inclusive.

Implication: These instructions are intended for operating system use. If this erratum occurs (and the OS does not ensure that the processor never has a Null segment selector in the SS or CS segment registers), the processor's behavior may become unpredictable, possibly resulting in system failure.

Workaround: Do not initialize the SYSTEM_CS_MSR with the values between FFF8h and FFFBh, FFF0h and FFF3h, or FFE8h and FFEBh before executing SYSENTER or SYSEXIT.

Status: For the steppings affected, see the Summary Tables of Changes.

AN42. Writing the Local Vector Table (LVT) When an Interrupt Is Pending May Cause an Unexpected Interrupt

Problem: If a local interrupt is pending when the LVT entry is written, an interrupt may be taken on the new interrupt vector even if the mask bit is set.

Implication: An interrupt may immediately be generated with the new vector when a LVT entry is written, even if the new LVT entry has the mask bit set. If there is no Interrupt Service Routine (ISR) set up for that vector the system will GP fault. If the ISR does not do an End of Interrupt (EOI) the bit for the vector will be left set in the in-service register and mask all interrupts at the same or lower priority.

Workaround: Any vector programmed into an LVT entry must have an ISR associated with it, even if that vector was programmed as masked. This ISR routine must do an EOI to clear any unexpected interrupts that may occur. The ISR associated with the spurious vector does not generate an EOI, therefore the spurious vector should not be used when writing the LVT.

Status: For the steppings affected, see the Summary Tables of Changes.

AN43. Using 2-M/4-M Pages When A20M# Is Asserted May Result in Incorrect Address Translations

Problem: An external A20M# pin if enabled forces address bit 20 to be masked (forced to zero) to emulates real-address mode address wraparound at 1 megabyte. However, if all of the following conditions are met, address bit 20 may not be masked.

- paging is enabled
- a linear address has bit 20 set
- the address references a large page
- A20M# is enabled

Implication: When A20M# is enabled and an address references a large page the resulting translated physical address may be incorrect. This erratum has not been observed with any commercially available operating system.

Workaround: Operating systems should not allow A20M# to be enabled if the masking of address bit 20 could be applied to an address that references a large page. A20M# is normally only used with the first megabyte of memory.

Status: For the steppings affected, see the Summary Tables of Changes.

AN44. Counter Enable bit [22] of IA32_CR_PerfEvtSel0 and IA32_CR_PerfEvtSel1 Do not Comply with PerfMon (Architectural Performance Monitoring) Specification

Problem: According to the Architectural Performance Monitoring specification the two PerfMon counters can be disabled/enabled through the corresponding Counter Enable bit [22] of IA32_CR_PerfEvtSel0/1.

Due to this erratum the following occurs:

1. bit [22] of IA32_CR_PerfEvtSel0 enables/disables both counters
2. bit [22] of IA32_CR_PerfEvtSel1 doesn't function

Implication: Software cannot enable/disable only one of the two PerfMon counters through the corresponding Counter Enable bit [22] of IA32_CR_PerfEvtSel0/1.

Workaround: Software should enable/disable both PerfMon counters together through Counter Enable bit [22] of IA32_CR_PerfEvtSel0 only. Alternatively, Software can effectively disable any one of the counters by clearing both Kernel and App bits [17:16] in the corresponding IA32_CR_PerfEvtSel0/1.

Status: For the steppings affected, see the Summary Tables of Changes.

AN45. Premature Execution of a Load Operation Prior to Exception Handler Invocation

Problem: If any of the below circumstances occur it is possible that the load portion of the instruction will have executed before the exception handler is entered.

1. If an instruction that performs a memory load causes a code segment limit violation
2. If a waiting floating-point instruction or MMX™ instruction that performs a memory load has a floating-point exception pending
3. If an MMX or SSE instruction that performs a memory load and has either CR0.EM=1 (Emulation bit set), or a floating-point Top-of-Stack (FP TOS) not equal to 0, or a DNA exception pending

Implication: In normal code execution where the target of the load operation is to write back memory there is no impact from the load being prematurely executed, nor from the restart and subsequent re-execution of that instruction by the exception handler. If the target of the load is to uncached memory that has a system side-effect, restarting the instruction may cause unexpected system behavior due to the repetition of the side-effect.

Workaround: Code which performs loads from memory that has side-effects can effectively workaround this behavior by using simple integer-based load instructions when accessing side-effect memory and by ensuring that all code is written such that a code segment limit violation cannot occur as a part of reading from side-effect memory.

Status: For the steppings affected, see the Summary Tables of Changes.

AN46. Performance Monitoring Events for Retired Instructions (C0H) May Not Be Accurate

Problem: The INST_RETIRED performance monitor may miscount retired instructions as follows:

- Repeat string and repeat I/O operations are not counted when a hardware interrupt is received during or after the last iteration of the repeat flow.
- VMLAUNCH and VMRESUME instructions are not counted.
- HLT and MWAIT instructions are not counted. The following instructions, if executed during HLT or MWAIT events, are also not counted:

  a) RSM from a C-state SMI during an MWAIT instruction.

  b) RSM from an SMI during a HLT instruction.

Implication: There may be a smaller than expected value in the INST_RETIRED performance monitoring counter. The extent to which this value is smaller than expected is determined by the frequency of the above cases.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN47. #GP Fault Is Not Generated on Writing IA32_MISC_ENABLE [34] When Execute Disable Bit Is Not Supported

Problem: #GP fault is not generated on writing to IA32_MISC_ENABLE [34] bit in a processor which does not support Execute Disable Bit functionality.

Implication: Writing to IA32_MISC_ENABLE [34] bit is silently ignored without generating a fault.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN48. Update of Read/Write (R/W) or User/Supervisor (U/S) or Present (P) Bits without TLB May Cause Unexpected Processor Behavior

Problem: Updating a page table entry by changing R/W, U/S or P bits without TLB shootdown (as defined by the 4 step procedure in "Propagation of Page Table and Page Directory Entry Changes to Multiple Processors" in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*), in conjunction with a complex sequence of internal processor micro-architectural events, may lead to unexpected processor behavior.

Implication: This erratum may lead to livelock, shutdown or other unexpected processor behavior. Intel has not observed this erratum with any commercially available system.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN49. SSE/SSE2 Streaming Store Resulting in a Self-Modifying Code (SMC) Event May Cause Unexpected Behavior

Problem: An SSE or SSE2 streaming store that results in a Self-Modifying Code (SMC) event may cause unexpected behavior. The SMC event occurs on a full address match of code contained in L1 cache.

Implication: Due to this erratum, any of the following events may occur:

1. A data access break point may be incorrectly reported on the instruction pointer (IP) just before the store instruction.

2. A non-cacheable store can appear twice on the external bus (the first time it will write only 8 bytes, the second time it will write the entire 16 bytes).

Note: Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN50. Shutdown Condition May Disable Non-Bootstrap Processors

Problem: When a logical processor encounters an error resulting in shutdown, non-bootstrap processors in the package may be unexpectedly disabled.

Implication: Non-bootstrap logical processors in the package that have not observed the error condition may be disabled and may not respond to INIT#, SMI#, NMI#, SIPI or other events.

Workaround: When this erratum occurs, RESET# must be asserted to restore multi-core functionality.

Status: For the steppings affected, see the Summary Tables of Changes.

AN51. Split Locked Stores May Not Trigger the Monitoring Hardware

Problem: Logical processors normally resume program execution following the MWAIT, when another logical processor performs a write access to a WB cacheable address within the address range used to perform the MONITOR operation. Due to this erratum, a logical processor may not resume execution until the next targeted interrupt event or O/S timer tick following a locked store that spans across cache lines within the monitored address range.

Implication: The logical processor that executed the MWAIT instruction may not resume execution until the next targeted interrupt event or O/S timer tick in the case where the monitored address is written by a locked store which is split across cache lines.

Workaround: Do not use locked stores that span cache lines in the monitored address range.

Status: For the steppings affected, see the Summary Tables of Changes.

AN52. Writing Shared Unaligned Data That Crosses a Cache Line without Proper Semaphores or Barriers May Expose a Memory Ordering Issue

Problem: Software which is written so that multiple agents can modify the same shared unaligned memory location at the same time may experience a memory ordering issue if multiple loads access this shared data shortly thereafter. Exposure to this problem requires the use of a data write which spans a cache line boundary.

Implication: This erratum may cause loads to be observed out of order. Intel has not observed this erratum with any commercially available software or system.

Workaround: Software should ensure at least one of the following is true when modifying shared data by multiple agents:

- The shared data is aligned
- Proper semaphores or barriers are used in order to prevent concurrent data accesses

Status: For the steppings affected, see the Summary Tables of Changes.

AN53. MSRs Actual Frequency Clock Count (IA32_APERF) or Maximum Frequency Clock Count (IA32_MPERF) May Contain Incorrect Data after a Machine Check Exception (MCE)

Problem: When an MCE occurs during execution of a RDMSR instruction for MSRs Actual Frequency Clock Count (IA32_APERF) or Maximum Frequency Clock Count (IA32_MPERF), the current and subsequent RDMSR instructions for these MSRs may contain incorrect data.

Implication: After an MCE event, accesses to the IA32_APERF and IA32_MPERF MSRs may return incorrect data. A subsequent reset will clear this condition.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN54. Using Memory Type Aliasing with Memory Types WB/WT May Lead to Unpredictable Behavior

Problem: Memory type aliasing occurs when a single physical page is mapped to two or more different linear addresses, each with different memory type. Memory type aliasing with the memory types WB and WT may cause the processor to perform incorrect operations leading to unpredictable behavior.

Implication: Software that uses aliasing of WB and WT memory types may observe unpredictable behavior. Intel chipset-based platforms are not affected by this erratum.

Workaround: None identified. Intel does not support the use of WB and WT page memory type aliasing.

Status: For the steppings affected, see the Summary Tables of Changes.

AN55. Code Breakpoint May be Taken after POP SS Instruction If It is Followed by an Instruction That Faults

Problem: A POP SS instruction should inhibit all interrupts including Code Breakpoints until after execution of the following instruction. This allows sequential execution of POP SS and MOV eSP, eBP instructions without having an invalid stack during interrupt handling. However, a code breakpoint may be taken after POP SS if it is followed by an instruction that faults. This results in a code breakpoint being reported on an unexpected instruction boundary since both instructions should be atomic.

Implication: This can result in a mismatched Stack Segment and SP. Intel has not observed this erratum with any commercially available software, or system.

Workaround: As recommended in the *Intel® 64 and IA-32 Architectures Software Developer's Manual*, the use "POP SS" in conjunction with "MOV eSP, eBP" will avoid the failure since the "MOV" will not fault.

Status: For the steppings affected, see the Summary Tables of Changes.

AN56. Incorrect Address Computed for Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update

Problem: A partial memory state save of the 512-byte FXSAVE image or a partial memory state restore of the FXRSTOR image may occur if a memory address exceeds the 64KB limit while the processor is operating in 16-bit mode or if a memory address exceeds the 4GB limit while the processor is operating in 32-bit mode.

Implication: FXSAVE/FXRSTOR will incur a #GP fault due to the memory limit violation as expected but the memory state may be only partially saved or restored.

Workaround: Software should avoid memory accesses that wrap around the respective 16-bit and 32-bit mode memory limits.

Status: For the steppings affected, see the Summary Tables of Changes.

AN57. Values for LBR/BTS/BTM Will Be Incorrect after an Exit from SMM

Problem: After a return from SMM (system management mode), the CPU will incorrectly update the LBR (last branch record) and the BTS (branch trace store), hence rendering their data invalid. The corresponding data if sent out as a BTM on the system bus will also be incorrect. Note: This issue would only occur when one of the 3 above mentioned debug support facilities are used.

Implication: The value of the LBR, BTS, and BTM immediately after an RSM operation should not be used.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN58. EFLAGS Discrepancy on a Page Fault after a Multiprocessor TLB Shootdown

Problem: This erratum may occur when the processor executes one of the following read-modify-write arithmetic instructions and a page fault occurs during the store of the memory operand: ADD, AND, BTC, BTR, BTS, CMPXCHG, DEC, INC, NEG, NOT, OR, ROL/ROR, SAL/SAR/SHL/SHR, SHLD, SHRD, SUB, XOR, and XADD. In this case, the EFLAGS value pushed onto the stack of the page fault handler may reflect the status of the register after the instruction would have completed execution rather than before it. The following conditions are required for the store to generate a page fault and call the operating system page fault handler:

1. The store address entry must be evicted from the DTLB by speculative loads from other instructions that hit the same way of the DTLB before the store has completed. DTLB eviction requires at least three-load operations that have linear address bits 15:12 equal to each other and address bits 31:16 different from each other in close physical proximity to the arithmetic operation.
2. The page table entry for the store address must have its permissions tightened during the very small window of time between the DTLB eviction and execution of the store. Examples of page permission tightening include from Present to Not Present or from Read/Write to Read Only, etc.
3. Another processor, without corresponding synchronization and TLB flush, must cause the permission change.

Implication: This scenario may only occur on a multiprocessor platform running an operating system that performs "lazy" TLB shootdowns. The memory image of the EFLAGS register on the page fault handler's stack prematurely contains the final arithmetic flag values although the instruction has not yet completed. Intel has not identified any operating systems that inspect the arithmetic portion of the EFLAGS register during a page fault nor observed this erratum in laboratory testing of software applications.

Workaround: No workaround is needed upon normal restart of the instruction, since this erratum is transparent to the faulting code and results in correct instruction behavior. Operating systems may ensure that no processor is currently accessing a page that is scheduled to have its page permissions tightened or have a page fault handler that ignores any incorrect state.

Status: For the steppings affected, see the Summary Tables of Changes.

AN59. Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior

Problem: Returning back from SMM mode into real mode while EFLAGS.VM is set in SMRAM may result in unpredictable system behavior.

Implication: If SMM software changes the values of the EFLAGS.VM in SMRAM, it may result in unpredictable system behavior. Intel has not observed this behavior in commercially available software.

Workaround: SMM software should not change the value of EFLAGS.VM in SMRAM.

Status: For the steppings affected, see the Summary Tables of Changes.

AN60. A Thermal Interrupt Is Not Generated When the Current Temperature Is Invalid

Problem: When the DTS (Digital Thermal Sensor) crosses one of its programmed thresholds it generates an interrupt and logs the event (IA32_THERM_STATUS MSR (019Ch) bits [9,7]). Due to this erratum, if the DTS reaches an invalid temperature (as indicated IA32_THERM_STATUS MSR bit[31]) it does not generate an interrupt even if one of the programmed thresholds is crossed and the corresponding log bits become set.

Implication: When the temperature reaches an invalid temperature the CPU does not generate a Thermal interrupt even if a programmed threshold is crossed.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN61. Performance Monitoring Event FP_ASSIST May Not Be Accurate

Problem: Performance monitoring event FP_ASSIST (11H) may be inaccurate as assist events will be counted twice per actual assist in the following specific cases:

- FADD and FMUL instructions with a not a Number (NaN) operand and a memory operand.
- FDIV instruction with zero operand value in memory

In addition, an assist event may be counted when DAZ (Denormals-Are-Zeros) and FTZ (Flush-To-Zero) flags are turned on even though no actual assist occurs.

Implication: The counter value for performance monitoring event FP_ASSIST (11H) may be larger than expected. The size of the error is dependent on the number of occurrences of the above condition while the event is active.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN62. The BS Flag in DR6 May Be Set for Non-Single-Step #DB Exception

Problem: DR6 BS (Single Step, bit 14) flag may be incorrectly set when the TF (Trap Flag, bit 8) of the EFLAGS Register is set and a #DB (Debug Exception) occurs due to one of the following:

- DR7 GD (General Detect, bit 13) being bit set;
- INT1 instruction;
- Code breakpoint

Implication: The BS flag may be incorrectly set for non-single-step #DB exception.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes

AN63. BTM/BTS Branch-From Instruction Address May Be Incorrect for Software Interrupts

Problem: When BTM (Branch Trace Message) or BTS (Branch Trace Store) is enabled, a software interrupt may result in the overwriting of BTM/BTS branch-from instruction address by the LBR (Last Branch Record) branch-from instruction address.

Implication: A BTM/BTS branch-from instruction address may get corrupted for software interrupts.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes

AN64. Store to WT Memory Data May Be Seen in Wrong Order by Two Subsequent Loads

Problem: When data of Store to WT memory is used by two subsequent loads of one thread and another thread performs cacheable write to the same address the first load may get the data from external memory or L2 written by another core, while the second load will get the data straight from the WT Store.

Implication: Software that uses WB to WT memory aliasing may violate proper store ordering.

Workaround: Do not use WB to WT aliasing.

Status: For the steppings affected, see the Summary Tables of Changes

AN65. Single Step Interrupts with Floating Point Exception Pending May Be Mishandled

Problem: In certain circumstances, when a floating point exception (#MF) is pending during single-step execution, processing of the single-step debug exception (#DB) may be mishandled.

Implication: When this erratum occurs, #DB will be incorrectly handled as follows

- #DB is signaled before the pending higher priority #MF (Interrupt 16)
- #DB is generated twice on the same instruction

Workaround: None identified

Status: For the steppings affected, see the Summary Tables of Changes

AN66. Fault on ENTER Instruction May Result in Unexpected Values on Stack Frame

Problem: The ENTER instruction is used to create a procedure stack frame. Due to this erratum, if execution of the ENTER instruction results in a fault, the dynamic storage area of the resultant stack frame may contain unexpected values (i.e. residual stack data as a result of processing the fault).

Implication: Data in the created stack frame may be altered following a fault on the ENTER instruction. Please refer to "Procedure Calls For Block-Structured Languages" in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Vol. 1, Basic Architecture*, for information on the usage of ENTER instructions. This erratum is not expected to occur in ring 3. Faults are usually processed in ring 0 and stack switch occurs when transferring to ring 0. Intel has not observed this erratum on any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes

AN67. Non-Temporal Data Store May Be Observed in Wrong Program Order

Problem: When non-temporal data is accessed by multiple read operations in one thread while another thread performs a cacheable write operation to the same address, the data stored may be observed in wrong program order (i.e. later load operations may read older data).

Implication: Software that uses non-temporal data without proper serialization before accessing the non-temporal data may observe data in wrong program order.

Workaround: Software that conforms to the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*, Section titled *Buffering of Write Combining Memory Locations*, will operate correctly.

Status: For the steppings affected, see the Summary Tables of Changes

AN68. Unaligned Accesses to Paging Structures May Cause the Processor to Hang

Problem: When an unaligned access is performed on paging structure entries, accessing a portion of two different entries simultaneously, the processor may live lock.

Implication: When this erratum occurs, the processor may live lock causing a system hang.

Workaround: Do not perform unaligned access on paging structure entries.

Status: For the steppings affected, see the Summary Tables of Changes

AN69. Microcode Updates Performed During VMX Non-root Operation Could Result in Unexpected Behavior

Problem: When Intel® Virtualization Technology is enabled, microcode updates are allowed only during VMX root operations. Attempts to apply microcode updates while in VMX non-root operation should be silently ignored. Due to this erratum, the processor may allow microcode updates during VMX non-root operations if not explicitly prevented by the host software.

Implication: Microcode updates performed in non-root operation may result in unexpected system behavior.

Workaround: Host software should intercept and prevent loads to IA32_BIOS_UPDT_TRIG MSR (79H) during VMX non-root operations. There are two mechanism that can be used (1) Enabling MSR access protection in the VM-execution controls or (2) Enabling selective MSR protection of IA32_BIOS_UPDT_TRIG MSR.

Status: For the steppings affected, see the Summary Tables of Changes

AN70. INVLPG Operations for Large (2-M/4-M) Pages May Be Incomplete under Certain Conditions

Problem: The INVLPG instruction may not completely invalidate Translation Look-aside Buffer (TLB) entries for large pages (2M/4M) when both of the following conditions exist:

- Address range of the page being invalidated spans several Memory Type Range Registers (MTRRs) with different memory types specified
- INVLPG operation is preceded by a Page Assist Event (Page Fault (#PF) or an access that results in either A or D bits being set in a Page table Entry (PTE))

Implication: Stale Translations may remain valid in TLB after a PTE update resulting in unpredictable system behavior. Intel has not observed this erratum with any commercially available software.

Workaround: Software should ensure that the memory type specified in the MTRRs is the same for the entire address range of the large page.

Status: For the steppings affected, see the Summary Tables of Changes

AN71. Page Access Bit May Be Set Prior to Signaling a Code Segment Limit Fault

Problem: If code segment limit is set close to the end of a code page, then due to this erratum the memory page Access bit (A Bit) may be set for the subsequent page prior to general protection fault on code segment limit.

Implication: When this erratum occurs, a non-accessed page, which is present in memory and follows a page that contains the code segment limit may be tagged as accessed.

Workaround: Erratum can be avoided by placing a guard page (non-present or non-executable page) as the last page of the segment or after the page that includes the code segment limit.

Status: For the steppings affected, see the Summary Tables of Changes

AN72. Performance Monitoring Events for Hardware Prefetch Requests (4EH) and Hardware Prefetch Request Cache Misses (4FH) May Not Be Accurate

Problem: Performance monitoring events that count hardware prefetch requests and prefetch misses may not be accurate.

Implication: This erratum may cause inaccurate counting for Hardware Prefetch Requests and Hardware Prefetch Request Cache Misses.

Workaround: Erratum can be avoided by placing a guard page (non-present or non-executable page) as the last page of the segment or after the page that includes the code segment limit.

Status: For the steppings affected, see the Summary Tables of Changes

AN73. EFLAGS, CR0, CR4 and the EXF4 Signal May Be Incorrect after Shutdown

Problem: When the processor is going into shutdown due to an RSM inconsistency failure, EFLAGS, CR0 and CR4 may be incorrect. In addition the EXF4 signal may still be asserted. This may be observed if the processor is taken out of shutdown by NMI#

Implication: A processor that has been taken out of shutdown may have an incorrect EFLAGS, CR0 and CR4. In addition the EXF4 signal may still be asserted.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AN74. An Asynchronous MCE during a Far Transfer May Corrupt ESP

Problem: If an asynchronous machine check occurs during an interrupt, call through gate, FAR RET or IRET and in the presence of certain internal conditions, ESP may be corrupted.

Implication: If the MCE (Machine Check Exception) handler is called without a stack switch, then a triple fault will occur due to the corrupted stack pointer, resulting in a processor shutdown. If the MCE is called with a stack switch, e.g. when the CPL (Current Privilege Level) was changed or when going through an interrupt task gate, then the corrupted ESP will be saved on the new stack or in the TSS (Task State Segment), and will not be used.

Workaround: Use an interrupt task gate for the machine check handler.

Status: For the steppings affected, see the Summary Tables of Changes.

AN75. Store Ordering May Be Incorrect between WC and WP Memory Types

Problem: According to the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A, Methods of Caching Available*, WP (Write Protected) stores should drain the WC (Write Combining) buffers in the same way as UC (Uncacheable) memory type stores do. Due to this erratum, WP stores may not drain the WC buffers.

Implication: Memory ordering may be violated between the WC and WP stores.

Workaround: None Identified

Status: For the steppings affected, see the Summary Tables of Changes

§

Specification Changes

There are no specification changes in this specification update revision.

§

Specification Clarifications

There are no specification clarifications in this specification update revision.

§

Documentation Changes

There are no documentation changes in this specification update revision.

Documentation changes for the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volumes 1, 2A, 2B, 3A and 3B* will be posted in a separate document, *Intel® 64 and IA-32 Architectures Optimization Reference Manual Documentation Changes*. Follow the link below to become familiar with this file.

http://developer.intel.com/design/processor/specupdt/252046.htm

§